UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/808,781	03/15/2001	Dov Dori	P-7481-US	5493

7590 03/02/2005

Eitan, Pearl, latzer & Cohen Zedek, LLP
10 Rockefeller Plaza
Suite 1001
New York, NY 10020

EXAMINER	
PROCTOR, JASON SCOTT	

ART UNIT	PAPER NUMBER
2123	

DATE MAILED: 03/02/2005

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No.	Applicant(s)	
	09/808,781	DORI, DOV	
	Examiner	Art Unit	
	Jason Proctor	2123	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *27 December 2004*.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-40* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-40* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on *27 October 2004* is/are: a)☒ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other: _____.

DETAILED ACTION

Claims 1-34 have been previously rejected. Claims 1-3, 5, 8, 12, 13, 15-18, 20, 24, 25, and 30-34 have been amended. New claims 35-40 have been added.

Claims 1-34 have been resubmitted for examination in light of Applicant's arguments and amendments. Claims 35-40 have been submitted for examination.

Claims 1-40 have been rejected.

Oath/Declaration

The Examiner thanks Applicant for resubmitting the signed Oath/Declaration. This paper has been recorded in the file.

Drawings

The Examiner thanks Applicant for submission of replacement drawing sheets that address the objections in the previous office action. Those objections to the drawings have been withdrawn.

Specification

The Examiner thanks Applicant for amending the specification to address the informalities objected to in the previous office action. Those objections to the specification have been withdrawn.

Claim Objections

The Examiner thanks Applicant for amending the claims to address the objections in the previous office action. Those objections to the claims have been withdrawn.

Claim Rejections – 35 U.S.C. § 112, first paragraph

The Examiner thanks Applicant for amending the specification to address the written description rejection of the previous office action. The Examiner concurs that in accordance with MPEP 2163.06, no new matter has been added. The written description rejection of the previous office action has been withdrawn.

Claim Rejections – 35 U.S.C. § 112, second paragraph

The Examiner thanks Applicant for amending the claims to overcome the rejections under 35 U.S.C. § 112, second paragraph, in the previous office action. Those rejections have been withdrawn.

Claim Interpretations

Regarding the claim interpretations of the previous office action, Applicant argued:

> Applicant respectfully submits that the interpretations made by the Examiner have no bearing on this Amendment and do not relate to any of the issues discussed herein. Therefore, although Applicant has not analyzed the appropriateness of the claim interpretations made by the Examiner, Applicant believes that this Amendment is fully responsive to all issues raised by the Examiner.

Applicant admits to having performed no analysis of the appropriateness of the Examiner's claim interpretations. The Examiner does believe those interpretations have

bearing on this Amendment and are relevant to the issues discussed therein. Accordingly, those interpretations are maintained.

Claim Rejections – 35 U.S.C. § 102(b)

Regarding the rejection of claims 1-11, 14, and 33 under 35 U.S.C. § 102(b) as being anticipated by US Patent No. 5,187,788 to Marmelstein, Applicant argued:

> Amended independent claim 1 recites, *inter alia*, "identifying a graphical pattern corresponding to a combination of one or more processes and one or more objects in said model diagram". Marmelstein does not disclose, teach, or suggest at least these features of claim 1.
> Amended independent claim 33 recites, *inter alia*, "identifying a graphical pattern corresponding to a combination of one or more processes and one or more objects in said model diagram". Marmelstein does not disclose, teach, or suggest at least these features of claim 33.

The Examiner respectfully traverses this argument as follows.

Marmelstein teaches that Ada code is generated from the graphical representation (column 2, lines 58-63). Marmelstein teaches that there is "a direct mapping between the graphical icons and the language constructs themselves" (column 20, lines 53-65). Marmelstein also teaches a command that causes Ada code for the "current APEX system representation" to be generated (column 18, lines 53-57). As can be seen from Figs. 5-8, which are screen dumps of APEX displays for a sample system file (column 4, lines 5-8), the APEX system representation is a graphical representation of a system, equivalent to a *model diagram*, wherein components of the graphical representation include *objects*, for example "Sensors" or "Controller", and *processes*, for example "Begin_Operation" or "Check_for_Coin", all found in Fig. 6.

It is clearly the graphical pattern of the model diagram in Marmelstein that directs the generation of Ada program code. It is clearly inherent that in order to do so, the

invention disclosed by Marmelstein must identify the graphical patterns in the model diagram for which Ada program code is being generated.

Regarding dependent claims 1-10 and 14, Applicant argued:

> Claims 2-11 and 14 are dependent from amended independent claim 1, and include all the features of amended independent claim 1 as well as additional distinguishing features. Therefore, it is respectfully submitted that the novelty of claims 2-11 and 14 follows directly from the novelty of amended independent claim 1 and therefore, none of claims 2-11 and 14 is anticipated by Marmelstein.

Applicant has not argued any specific "additional distinguishing features" of claims 2-11 and 14. Therefore, in traversing Applicant's arguments regarding independent claim 1, the Examiner has traversed Applicant's arguments regarding dependent claims 2-11 and 14.

Regarding the rejection of claims 16-29 and 34 under 35 U.S.C. § 102(b) as being anticipated by US Patent No. 4,315,315 to Kossiakoff, applicant argued:

> Amended independent claim 16 recites, *inter alia*, "identifying a graphical pattern corresponding to a combination of one or more processes and one or more objects in said model diagram". Kossiakoff does not disclose, teach, or suggest at least these features of claim 16.
> Amended independent claim 34 recites, *inter alia*, "identifying a graphical pattern corresponding to a combination of one or more processes and one or more objects in said model diagram". Kossiakoff does not disclose, teach, or suggest at least these features of claim 1.

The Examiner respectfully traverses this argument as follows.

Kossiakoff teaches that a Transformation Program "converts the data flow circuit", which is the subject being graphically modeled (column 4, lines 41-52), into an operational sequence. In the next step, the computer converts the operational sequence into computer assembly code (column 4, line 58 – column 5, line 5, section titled "Transformation of Graphical into Logical Form").

It is clearly the graphical pattern of the model diagram in Kossiakoff that directs the generation of computer assembly code. It is clearly inherent that in order to do so, the invention disclosed by Kossiakoff must identify the graphical patterns in the model diagram for which computer assembly code is being generated.

Claim Rejections - 35 USC § 103

Regarding the rejections of claims 12-13 as being unpatentable over Marmelstein in view of US Patent No. 5,321,607 to Fukumochi, Applicant argued:

> Claims 12 and 13 depend from amended independent claim 1, which recites, *inter alia*, "identifying a graphical pattern corresponding to a combination of one or more processes and one or more objects in said model diagram". Marmelstein and/or Fukumochi, alone or in combination, do not disclose, teach, or suggest at least this feature [of] the claimed invention.

The Examiner respectfully traverses this argument by referring above to the discussion of Marmelstein and asserts that Marmelstein does at least suggest this limitation. Therefore, the combination of Marmelstein and Fukumochi does indeed render claims 12 and 13 obvious.

Further regarding the relevance of Marmelstein as prior art, Applicant argued:

> Generally, the claimed invention is directed to, *inter alia*, a method and device for modeling a system, for example, a social system, a biological system, a physical system, or an informatical system. In contrast, Marmelstein is directed to an automatic code generation tool for the Ada programming language. The characteristics of the field of modeling systems are significantly different from those of the field of automatic code generation. Therefore, Marmelstein may not be properly used as relevant prior art to render the claimed invention obvious.

The Examiner respectfully traverses this argument as follows:

The graphical programming tool taught by Marmelstein is a modeling system. Indeed, Marmelstein makes several references to the system as a model (column 14,

lines 44-57; column 15, lines 34-41). While the intended use of Applicant's invention may be broader, this is insufficient to render of the teachings of Marmelstein irrelevant as prior art. Therefore, the teachings of Marmelstein are appropriate prior art.

Further regarding the relevance of Fukumochi as prior art, Applicant argued:

> As discussed above, the claimed invention is directed to, *inter alia*, a method and device for modeling a system, for example, a social system, a biological system, a physical system, or an informatical system. In contrast, Fukumochi is directed to an automatic translation machine for translation between two natural languages, and is clearly unrelated to modeling or to diagrams having graphical elements.

The Examiner respectfully traverses this argument as follows:

Amended claim 12 recites the limitation "translating a label of a graphic element from a subset of said first natural language to a subset of a second natural language." MPEP 2141.01(a) reads as follows:

> The examiner must determine what is "analogous prior art" for the purpose of analyzing the obviousness of the subject matter at issue. "In order to rely on a reference as a basis for rejection of an applicant's invention, the reference must either be in the field of applicant's endeavor or, if not, then be reasonably pertinent to the particular problem with which the inventor was concerned." *In re Oetiker*, 977 F.2d 1443, 1446, 24 USPQ2d 1443, 1445 (Fed. Cir. 1992).

The particular problem concerning claim 12 is translating labels from a subset of one natural language to a subset of a second natural language. As such, the teachings of Fukumochi, which relate to machine translation between two natural languages, is analogous prior art. Claim 13, which depends from claim 12, and is likewise concerned with translation of labels from a subset of one natural language to a subset of a second natural language.

Further regarding the combination of Marmelstein and Fukumochi, Applicant argued:

> Applicant respectfully submits that at the time the invention was made, it would not have been obvious to combine the teaching of Marmelstein with the teaching of Fukumochi, since these two references belong to different, non-related fields. Therefore, Marmelstein and Fukumochi may not be properly used in combination to render the claimed invention obvious.

The Examiner respectfully traverses this argument as follows:

MPEP 2143.01 reads as follows:

> "There are three possible sources for a motivation to combine references: the nature of the problem to be solved, the teachings of the prior art, and the knowledge of persons of ordinary skill in the art." *In re Rouffet*, 149 F.3d 1350, 1357, 47 USPQ2d 1453, 1457-58 (Fed. Cir. 1998)

Although Marmelstein and Fukumochi belong to different fields, the problems posed by language barriers have long been appreciated in numerous arts. The field of machine translation of natural languages attempts to address this problem in order to better facilitate other fields of endeavor, notably human-computer interfaces. Indeed, it would be difficult to substantiate that there exists no motivation to incorporate automatic machine translation of natural languages into any computer interface, because the utility of any such combination would be immediately obvious to a person of ordinary skill in the art. Therefore, the combination of Marmelstein and Fukumochi satisfies at least two of the three established sources for motivation to combine references: the nature of the problem to be solved and the knowledge of persons of ordinary skill in the art.

Further regarding the combination of Marmelstein and Fukumochi, Applicant argued:

> Applicant submits that the combination of the code generator of Marmelstein with the natural language translator of Fukumochi would yield an inoperable device. [...] The translation machine of Fukumochi analyses a text to produce a text, whereas the code generation machine of Marmelstein produces an Ada code based on a graphical flow-chart. The translation machine of Fukumochi cannot analyze a graphical flow-chart and cannot produce an Ada code, whereas the code generation machine of Marmelstein cannot analyze a text and cannot produce a natural language text.

The Examiner respectfully traverses this argument as follows:

The Examiner respectfully submits that Applicant has mischaracterized the combination intended by the Examiner. Applicant's invention, at one stage of operation, consists of a model with labels in a first natural language, and upon application of a translation function, translates the labels of the model to a second natural language. Likewise, the combination formed by the Examiner in the rejection of claims 12-13 consists of Marmelstein's model with labels in a first natural language (Fig. 6, "Sensors", "Controller", etc.) and the translation function taught by Fukumochi to translate those labels into a second natural language. In the Examiner's combination, the input for the translation machine of Fukumochi is text (model labels in a first language) and the output is text (model labels in a second language). Therefore the combination is operable.

Regarding the rejection of claim 15 as being unpatentable over Marmelstein in view of Kossiakoff, Applicant argued:

> Claim 15 depends from amended independent claim 1, which recites, *inter alia*, "identifying a graphical pattern corresponding to a combination of one or more processes and one or more objects in said model diagram". Marmelstein and/or Kossiakoff, alone or in combination, do not disclose, teach, or suggest at least this feature [of] the claimed invention.

The Examiner respectfully traverses this argument by referring above to the discussion of Marmelstein and asserts that Marmelstein does at least suggest this limitation. Therefore, the combination of Marmelstein and Kossiakoff does indeed render claim 15 obvious.

Further regarding the relevance of Kossiakoff as prior art, Applicant argued:

> As discussed above, the claimed invention is directed to, *inter alia*, a method and device for modeling a system, for example, a social system, a biological system, a physical system, or an informatical system. In contrast, Kossiakoff is directed to a machine for producing assembly language code from a graphical circuit. The characteristics of the field of modeling systems are significantly different from those of the field of assembly code generation. Therefore, Kossiakoff may not be properly used as relevant prior art to render the claimed invention obvious.

The Examiner respectfully traverses this argument as follows:

The invention taught by Kossiakoff produces assembly code "directly from a two-dimensional network representing the flow of data and control logic which it is desired to accomplish on a general purpose computer". Such a representation clearly constitutes a model. While the intended use of Applicant's invention may be broader, this is insufficient to render of the teachings of Kossiakoff irrelevant as prior art. Therefore, the teachings of Kossiakoff are appropriate prior art.

Further regarding the combination of Marmelstein and Kossiakoff, Applicant argued:

> The machine of Marmelstein creates ADA code from a graphical representation prepared by a programmer operating a specific design tool, whereas the machine of Kossiakoff analyses a circuit to produce assembly code. The input, the output, and the operational mechanism of the Ada code generation machine of Marmelstein are significantly different from the input, the output, and the operational mechanism of the assembly production machine of Kossiakoff, respectively. Therefore, at the time the invention was made, there was no motivation or suggestion in the art to combine the machine of Marmelstein with the machine of Kossiakoff.

The Examiner respectfully traverses this argument as follows:

Marmelstein and Kossiakoff are both tools for graphical programming. Marmelstein and Kossiakoff both enable the user to graphically model a program being designed. Marmelstein and Kossiakoff both produce program code based on the graphical model. Marmelstein and Kossiakoff both exhibit certain advantages. Combining the known advantages of prior art without producing new or unexpected

results is obvious. In this case, the claim limitations were known in the art, as shown by Marmelstein and Kossiakoff, two closely related examples of the prior art.

Further regarding the combination of Marmelstein and Kossiakoff, Applicant argued:

> Applicant submits that the combination of the machine of Marmelstein with the machine of Kossiakoff would yield an inoperable device. The machine of Marmelstein receives as input, and processes, only a certain, unique type of graphical representation, not compatible with the type of graphical representation that the machine of Kossiakoff receives as input and processes, and vice versa.

The Examiner respectfully traverses this argument as follows:

The Examiner respectfully submits that Applicant has mischaracterized the combination intended by the Examiner. The combination formed by the Examiner relates to the combined teachings of the prior art, not a physical or direct functional combination of the disclosed inventions. The relevant teaching of Kossiakoff shows that the ability to simulate operation of the model is possible, has been implemented in the prior art, and teaches the advantages of that feature. The Examiner's rejection of claim 15 combines *that teaching* with Marmelstein, not the physical incorporation of Kossiakoff's invention with Marmelstein. As such, the Examiner asserts that the combination would yield an operable device.

Regarding claim 25, Applicant submits arguments that refer directly to the arguments of claim 15. The Examiner has addressed the arguments of claim 15 above and considers the similar arguments of claim 25 traversed.

Regarding the rejections of claims 30-32 as being unpatentable over Marmelstein in view of Fukumochi, Applicant argued:

> Amended independent claim 30 recites, *inter alia*, "identifying a graphical pattern corresponding to a combination of one or more processes and one or more objects". Marmelstein and/or Fukumochi, alone or in combination, do not disclose, teach, or suggest at least this feature [of] the claimed invention. Therefore, the combination of Marmelstein and Fukumochi does not render claim 30 obvious.

The Examiner respectfully traverses this argument by referring above to the discussion of Marmelstein and asserts that Marmelstein does at least suggest this limitation. Therefore, the combination of Marmelstein and Fukumochi does indeed render claim 30 obvious.

Regarding dependent claims 31 and 32, Applicant argued:

> Claims 31-32 are dependent from amended independent claim 30, and include all the features of amended independent claim 30 as well as additional distinguishing features. Therefore, it is respectfully submitted that the patentability of claims 31-32 follow directly from the novelty of amended independent claim 30.

Applicant has not argued any specific "additional distinguishing features" of claims 31-32. Therefore, in traversing Applicant's arguments regarding independent claim 30, the Examiner has traversed Applicant's arguments regarding dependent claims 31-32.

Further regarding claims 30-32, Applicant submits arguments that refer directly to the arguments of claims 12 and 13. The Examiner has addressed the arguments of claims 12 and 13 above and considers the similar arguments of claims 30-32 traversed.

Claim Objections

New claims 35-40 improperly rely upon provisional applications incorporated by reference for essential subject matter. The essential subject matter relied upon must be added to the disclosure of the application. See MPEP 608.01(p) and 2163.07(b). Appropriate correction is required.

Claim Rejections - 35 USC § 102

1. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –
>
> (b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

2. Claim 1-11, 14, and 33 are rejected under 35 U.S.C. 102(b) as being anticipated by Marmelstein.

3. Regarding Claim 1, Marmelstein teaches a program code generation tool which

receives input from the user that specifying a program in the form of a graphical flowchart (column 2, lines 58-63),

presents a textual representation of the flowchart (column 2, line 63 - column 3, line 4),

is updated to accurately correspond to the flowchart (column 3, lines 38-40; column 10, lines 54-61),

represents objects using rectangles (column 17, lines 52-65; figure 5),

represents operations associated with the objects using circles (column 17, lines 59-65; figure 5; column 22, lines 25-41; figure 6),

produces a program code textual description of the flowchart (column 2, lines 59-63; figures 15-25; column 11, lines 10-36), and

can produce a graphical representation of a flowchart by reading the corresponding program code textual description of the model (column 3, lines 9-13; column 18, lines 36-49; column 27, lines 8-13).

Marmelstein teaches that Ada code is generated from the graphical representation (column 2, lines 58-63). Marmelstein teaches that there is "a direct mapping between the graphical icons and the language constructs themselves" (column 20, lines 53-65). Marmelstein also teaches a command that causes Ada code for the "current APEX system representation" to be generated (column 18, lines 53-57). As can be seen from Figs. 5-8, which are screen dumps of APEX displays for a sample system file (column 4, lines 5-8), the APEX system representation is a graphical representation of a system, equivalent to a *model diagram*, wherein components of the graphical representation include *objects*, for example "Sensors" or "Controller", and *processes*, for example "Begin_Operation" or "Check_for_Coin", all found in Fig. 6.

4. Regarding Claim 2, Marmelstein teaches a program code generation tool wherein the tool models objects as rectangles in the APEX Editor (column 17, lines 52-65; figure 5) and operations associated with the objects are independently represented

as circles in the REM-Net Editor (column 17, lines 59-65; figure 5; column 22, lines 25-41; figure 6).

5. Regarding Claim 3, Marmelstein teaches a program code generation tool wherein the tool models objects as rectangles in the APEX Editor (column 17, lines 52-65; figure 5) and operations associated with the objects are independently represented as circles in the REM-Net Editor (column 17, lines 59-65; figure 5; column 22, lines 25-41; figure 6).

6. Regarding Claim 4, Marmelstein teaches a program code generation tool wherein the tool produces an ADA programming language textual description of a flowchart (column 2, lines 59-63; figures 15-25; column 11, lines 10-36). The use of context-free grammars to define programming language syntax is well known, indeed this was the first application of context-free grammars (See Sebesta, pages 109-110). A context-free grammar for ADA is known (See The Ada 95 Reference Manual, Annex P). Thus generating an ADA programming language textual description is deemed equivalent to generating a context-free grammar expression.

7. Regarding Claim 5, Marmelstein teaches a program code generation tool wherein the tool produces an ADA programming language textual description of a flowchart (column2, lines 59-63; figures 15-25; column 11, lines 10-36). ADA is a high level programming language (See Microsoft Computer Dictionary, Fifth Edition). Further, statements in a high level programming language generally use keywords similar to English (See Microsoft Computer Dictionary, Fifth Edition). Thus Marmelstein

teaches generating a textual description of a flowchart according to a context-free grammar that uses vocabulary from English, a natural language.

8. Regarding Claim 6, Marmelstein teaches displaying both the flowchart and the corresponding textual description simultaneously (figures 5-8; column 2, line 63 – column 3, line 9; column 4, lines 5-8).

9. Regarding Claim 7, Marmelstein teaches a user giving input to the program code generation tool (column 6, line 64 – column 7, line 6)

10. Regarding Claim 8, Marmelstein teaches a program code generation tool where the textual description corresponding to the flowchart is synchronously updated to reflect changes the user enacts (column 2, line 59 – column 3, line 4; column 3, lines 38-40; column 10, lines 54-61)

11. Regarding Claim 9, Marmelstein teaches a program code generation tool where the textual description corresponding to the flowchart can be generated in its entirety in response to a user's command (column 11, lines 10-15; column 11, lines 44-48; column 18, lines 53-57; figure 5).

12. Regarding Claim 10, Marmelstein teaches a program code generation tool where the user can specify a level of detail to depict by giving input into the APEX editor (column 6, lines 36-46) or the DS editor (column 6, lines 47-62). Also, the APEX editor permits the user to represent the program's top level package structure (column 17, lines 52-69) while certain user input will cause the program to depict a higher or lower level of detail (column 17, line 59 – column 18, line 2).

13. Regarding Claim 11, Marmelstein teaches a program code generation tool where the Data Structure Editor allows the user to provide input that causes the tool to display greater detail regarding the ADA program code textual description of the flowchart. The Data Structure Editor displays certain ADA program code such as data structures, subprograms, and variables in the scope of the current data structure (column 24, lines 43-51; column 25, lines 3-16).

14. Regarding Claim 14, Marmelstein teaches a program code generation tool which generates program code to implement the flowchart (column 2, lines 59-63; figures 15-25; column 11, lines 10-36).

15. Regarding Claim 33, Marmelstein teaches a computer program product which

receives input from the user that specifying a program in the form of a graphical flowchart (column 2, lines 58-63),

presents a textual representation of the flowchart (column 2, line 63 - column 3, line 4),

which is updated to accurately correspond to the flowchart (column 3, lines 38-40; column 10, lines 54-61),

the tool represents objects using rectangles (column 17, lines 52-65; figure 5),

the tool represents operations associated with the objects using circles (column 17, lines 59-65; figure 5; column 22, lines 25-41; figure 6),

the tool produces a program code textual description of the flowchart (column 2, lines 59-63; figures 15-25; column 11, lines 10-36), and

can produce a graphical representation of a flowchart by reading the corresponding program code textual description of the model (column 3, lines 9-13; column 18, lines 36-49; column 27, lines 8-13).

Marmelstein teaches that Ada code is generated from the graphical representation (column 2, lines 58-63). Marmelstein teaches that there is "a direct mapping between the graphical icons and the language constructs themselves" (column 20, lines 53-65). Marmelstein also teaches a command that causes Ada code for the "current APEX system representation" to be generated (column 18, lines 53-57). As can be seen from Figs. 5-8, which are screen dumps of APEX displays for a sample system file (column 4, lines 5-8), the APEX system representation is a graphical representation of a system, equivalent to a *model diagram*, wherein components of the graphical representation include *objects*, for example "Sensors" or "Controller", and *processes*, for example "Begin_Operation" or "Check_for_Coin", all found in Fig. 6.

16. Claims 16-29 and 34 are rejected under 35 U.S.C. 102(b) as being anticipated by Kossiakoff.

17. Regarding Claim 16, Kossiakoff teaches a method of modeling through the use of a computer program wherein

text input is received from a user to specify an electronic circuit diagram (column 4, lines 59-62; column 24, lines 42-50),

the circuit diagram being composed of graphic elements (column 3, lines 5-10; column 4, lines 42-52),

where polygons represent data circuit elements (figure 3; column 5, lines 37-38; column 3, lines 26-35), the objects of the model, and

solid and dashed lines represent the flow of data and control signals (column 6, lines 28-35), the processes of the model.

Further, the invention disclosed in Kossiakoff creates program code corresponding to the electronic circuit diagram, thereby modeling the operation of the electronic circuit, so that the user may detect and correct errors (column 4, line 63 – column 5, line 5).

Kossiakoff teaches that a Transformation Program "converts the data flow circuit", which is the subject being graphically modeled (column 4, lines 41-52), into an operational sequence. In the next step, the computer converts the operational sequence into computer assembly code (column 4, line 58 – column 5, line 5, section titled "Transformation of Graphical into Logical Form").

18. Regarding Claim 17, Kossiakoff teaches a method of modeling through the use of a computer program where the data flow circuit elements represent hardware building

blocks (column 7, lines 35-43), thereby modeling objects in the circuit. In the diagram, solid lines represent the flow of data while dashed lines represent control signals (column 6, lines 28-35), thereby modeling processes within the circuit.

19. Regarding Claim 18, Kossiakoff teaches a method of modeling through the use of a computer program where objects and processes are graphically represented as individual elements (figures 2-3; column 6, lines 28-35; column 7, lines 35-43).

20. Regarding Claim 19, Kossiakoff teaches a method of modeling through the use of a computer program that receives input from a user specifying an electronic circuit diagram (column 4, lines 59-62; column 24, lines 42-50). It is deemed inherent that the text input forming the computer-human interface consists of a context-free grammar. The understanding and use of natural languages is a much harder problem for computers than understanding computer languages (See TECHNICAL, "natural language").

21. Regarding Claim 20, Kossiakoff teaches a method of modeling through the use of a computer program that receives input from a user specifying an electronic circuit diagram (column 4, lines 59-62; column 24, lines 42-50). It is deemed inherent that the text input forming the computer-human interface consists of a context-free grammar. It is deemed inherent that the design of a context-free grammar for use as a computer-human interface would consist of vocabulary from the natural language of the intended user.

22. Regarding Claim 21, Kossiakoff teaches a method of modeling through the use of a computer program which receives input from a user specifying an electronic circuit

diagram (column 4, lines 59-62; column 24, lines 42-50). It is deemed inherent that the method disclosed requires parsing the received input in according to the context-free grammar recognized by the program.

23. Regarding Claim 22, Kossiakoff teaches that an object of the invention is to provide an alpha-numerically documented representation of the process to be modeled such that it is clearly understandable by an engineer, who represents the domain expert (column 3, lines 53-59).

24. Regarding Claim 23, Kossiakoff teaches that the user provides input to specify the electronic circuit diagram (column 4, lines 59-62; column 24, lines 42-50).

25. Regarding Claim 24, Kossiakoff teaches that the user may use a light pen to interactively create the diagram (column 12, lines 16-27).

26. Regarding Claim 26, Kossiakoff teaches that the circuit diagram may comprise "program blocks", which are themselves circuit diagrams (figures 6-7; column 22, lines 1-16; column 22, lines 32-39). The user manipulates these program blocks through the use of an Integration Editor, while the circuit diagrams represented by the program blocks are manipulated through the use of a separate editor (column 21, lines 56-69; column 12, lines 44-53). It is deemed inherent that the selection of either the data circuit editor or the Integration Editor is the result of user input.

27. Regarding Claim 27, Kossiakoff teaches that the circuit diagram may comprise "program blocks", which are themselves circuit diagrams (figures 6-7; column 22, lines 1-16; column 22, lines 32-39). The user manipulates these program blocks through the use of an Integration Editor, while the circuit diagrams represented by the program

blocks are manipulated through the use of a separate editor (column 21, lines 56-69; column 12, lines 44-53). It is deemed inherent that the selection of either the data circuit editor or the Integration Editor is the result of user input. These editors display different levels of detail in the diagram (column 5, lines 55-60; figures 7-8).

28. Regarding Claim 28, Kossiakoff teaches that the program automatically generates software instructions to implement the model (column 3, lines 26-35; column 4, line 63 – column 5, line 5; column 22, lines 32-39).

29. Regarding Claim 29, Kossiakoff teaches that the circuit diagram is converted into computer code and tested with sample inputs and outputs. The goal is to correct errors or omissions by the designer (column 4, line 66 – column 5, line 5). It is deemed that this constitutes a simulation of the modeled system.

30. Regarding Claim 34, Kossiakoff discloses a computer program product wherein

text input is received from a user to specify an electronic circuit diagram (column 4, lines 59-62; column 24, lines 42-50),

the circuit diagram being composed of graphic elements (column 3, lines 5-10; column 4, lines 42-52),

where polygons represent data circuit elements (figure 3; column 5, lines 37-38; column 3, lines 26-35), the objects of the model, and

solid and dashed lines represent the flow of data and control signals (column 6, lines 28-35), the processes of the model.

Further, the invention disclosed in Kossiakoff creates program code corresponding to the electronic circuit diagram, thereby modeling the

operation of the electronic circuit, so that the user may detect and correct errors (column 4, line 63 – column 5, line 5).

Kossiakoff teaches that a Transformation Program "converts the data flow circuit", which is the subject being graphically modeled (column 4, lines 41-52), into an operational sequence. In the next step, the computer converts the operational sequence into computer assembly code (column 4, line 58 – column 5, line 5, section titled "Transformation of Graphical into Logical Form").

Claim Rejections - 35 USC § 103

31. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

32. Claims 12 and 13 are rejected under 35 U.S.C. 103(a) as being unpatentable over Marmelstein as applied to claim 1 above, and further in view of Fukumochi et al.

33. Regarding Claim 12, Marmelstein does not teach the translation of labeled diagram elements from one natural language to another, however Fukumochi et al. discloses an automatic translating machine which makes use of parse trees (Fukumochi et al., column 3, line 58 – column 4, line 13). The combination of the translation device disclosed by Fukumochi et al. with the programming code generation tool disclosed by

Marmelstein could be realized as a feature initiated by user command to translate the labels of the various graphical elements. It is deemed that the advantages of a translating the labels of a flowchart representing a proposed program design into a second language are inherent in view of the various natural languages spoken by software developers. It would have been obvious to a person of ordinary skill in the art at the time of applicant's invention to incorporate the translation ability of Fukumochi et al. with the automatic diagramming and code generation ability of Marmelstein to improve the usability of the code generation tool.

34. Regarding Claim 13, the limitations of Claim 12 are rejected as above, and further Fukumochi et al. discloses the use of parse trees (column 3, line 58 – column 4, line 13) which are used to represent the syntactic structure of context-free grammars (Sebesta, 113). Additionally, Fukumochi et al. discloses the use of production rules which conform to a subset of grammatical rules for a natural language (column 2, Table 2; column 2, lines 28-35; column 6, lines 1-21).

35. Claim 15 is rejected under 35 U.S.C. 103(a) as being unpatentable over Marmelstein as applied to claim 1 above, and further in view of Kossiakoff.

36. Marmelstein does not teach simulating the flowchart developed using the program code generation tool. Kossiakoff does teach that the circuit diagram is converted into computer code and tested with sample inputs and outputs. The goal is to correct errors or omissions by the designer. (Kossiakoff, column 4, line 66 – column 5, line 5). It is deemed that this constitutes a simulation of the modeled system. Both Marmelstein and Kossiakoff disclose inventions that allow a user to graphically create a

diagram which is subsequently used to generate programming language code. The ability to simulate the graphical program as disclosed by Kossiakoff could have been implemented in the invention of Marmelstein with the addition of a final step of executing the generated program code with sample inputs and reporting the outputs to the user. It would have been obvious to a person of ordinary skill in the art to include features that aid the user to improve the quality of the generated code as this is a goal of both inventions (Marmelstein, column 3, lines 16-40; Kossiakoff, column 14, lines 36-46; column 5, lines 1-5).

37. Claim 25 is rejected under 35 U.S.C. 103(a) as being unpatentable over Kossiakoff as applied to claim 16 above, and further in view of Marmelstein.

38. Kossiakoff does not teach generating the diagram description in a batch mode, however Marmelstein teaches using the input of a complete ADA program package file and generating a complete diagram for that file (Marmelstein, column 3, lines 9-13; column 18, lines 36-49; column 27, lines 8-13). Such a feature could be implemented in the invention of Kossiakoff in an identical fashion without interfering with any existing functionality or goal of the invention. It would have been obvious to a person of ordinary skill in the art to recognize the advantage of generating a diagram from textual program code (Marmelstein, column 27, lines 15-23) and to incorporate said advantage in the invention of Kossiakoff to produce a tool with greater functionality.

39. Claims 30-32 are rejected under 35 U.S.C. 103(a) as being unpatentable over Marmelstein in view of Fukumochi et al.

40. Regarding Claim 30, Marmelstein teaches a computer program product which

receives input from the user that specifying a program in the form of a graphical flowchart (column 2, lines 58-63),

the elements of the flowchart labeled using a natural language (figure 5, elements 522, 524, 526; figure 6, elements 622, 623),

the tool represents objects using rectangles (column 17, lines 52-65; figure 5),

the tool represents operations associated with the objects using circles (column 17, lines 59-65; figure 5; column 22, lines 25-41; figure 6),

the tool presents a textual representation of the flowchart (column 2, line 63 - column 3, line 4), and

the tool produces a program code textual description of the flowchart (column 2, lines 59-63; figures 15-25; column 11, lines 10-36).

Marmelstein teaches that Ada code is generated from the graphical representation (column 2, lines 58-63). Marmelstein teaches that there is "a direct mapping between the graphical icons and the language constructs themselves" (column 20, lines 53-65). Marmelstein also teaches a command that causes Ada code for the "current APEX system representation" to be generated (column 18, lines 53-57). As can be seen from Figs. 5-8, which are screen dumps of APEX displays for a sample system file (column 4, lines 5-8), the APEX system representation is a graphical representation of a system, equivalent to a *model diagram*, wherein components of the graphical representation include *objects*, for

example "Sensors" or "Controller", and *processes*, for example "Begin_Operation" or "Check_for_Coin", all found in Fig. 6.

41. Fukumochi et al. teaches a computer program which translates text from one natural language to a second natural language (column 3, lines 46-51). The combination of the translation device disclosed by Fukumochi et al. with the programming code generation tool disclosed by Marmelstein could be realized as a feature initiated by user command to translate the labels of the various graphical elements. Doing so would facilitate the generation of a new textual program code wherein the labels in the second natural language would be represented in the program code. It is deemed that the advantages of a translating the labels of a flowchart representing a proposed program design into a second language are inherent in view of the various natural languages spoken by software developers. It would have been obvious to a person of ordinary skill in the art at the time of applicant's invention to incorporate the translation ability of Fukumochi et al. with the automatic diagramming and code generation ability of Marmelstein to improve the usability of the code generation tool.

42. Regarding Claim 31, the limitations of Claim 30 are rejected as above, further Marmelstein teaches a program code generation tool wherein the tool models objects as rectangles in the APEX Editor (column 17, lines 52-65; figure 5) and operations associated with the objects are independently represented as circles in the REM-Net Editor (column 17, lines 59-65; figure 5; column 22, lines 25-41; figure 6).

43. Regarding Claim 32, the limitations of Claim 31 are rejected as above, further Marmelstein teaches a program code generation tool wherein the tool produces an ADA programming language textual description of a flowchart (column 2, lines 59-63; figures 15-25; column 11, lines 10-36). The use of context-free grammars to define programming language syntax is well known; indeed this was the first application of context-free grammars (See Sebesta, pages 109-110). A context-free grammar for ADA is known (See The Ada 95 Reference Manual, Annex P). Thus generating an ADA programming language textual description is deemed equivalent to generating a context-free grammar expression.

44. Claim 35 is rejected under 35 U.S.C. § 103(a) as being unpatentable over Marmelstein as applied to claim 1 above, and further in view of US Patent No. 6,681,383 to Pastor et al. (Pastor).

45. Regarding claim 35, Marmelstein does not explicitly teach generating documentation of the model diagram.

Pastor teaches an automatic software production system that shares many common features with the automatic code generation machine of Marmelstein. Among these similarities are producing a model of the system (Figs. 3-4). Pastor also teaches a documentation generator that generates documentation of the system being modeled (column 36, lines 60-65). Pastor teaches that the format of the documentation uses HTML and Rich Text Format (column 37, lines 25-29). Official notice is taken that it is well known in the art that both HTML and Rich Text Format files can have graphics and

text. It would have been obvious to a person of ordinary skill in the art at the time of Applicant's invention to combine Pastor's teaching of the concept of a documentation generator with the automatic code generation machine of Marmelstein. This combination is not a physical combination of the two inventions, but rather a combination of the teachings found in the references. Motivation for combining a documentation generator with the automatic code generation machine of Marmelstein is readily found in the nature of the problem to be solved by an automatic code generation machine as well as the knowledge of persons of ordinary skill in the art.

46. Claims 36-40 are rejected under 35 U.S.C. § 103(a) as being unpatentable over Marmelstein as applied to claim 1 above, and further in view of US Patent No. 6,343,265 to Glebov et al. (Glebov).

Regarding claim 36, Marmelstein does not explicitly teach allowing a plurality of users to work in real time on said model diagram.

Glebov teaches a system for making a design model that shares many common features with the automatic code generation machine of Marmelstein. Among these similarities are producing a model of the system (Figs. 1 and 6). Glebov also teaches a distributed system including server objects capable of messaging and communication with other objects and a common repository located on a server. It would have been obvious to a person of ordinary skill in the art at the time of Applicant's invention that the distributed architecture taught by Glebov enables a plurality of users to work on a model in real time. It would have been obvious to a person of ordinary skill in the art at the

time of Applicant's invention to combine the concept of a distributed architecture with the automatic code generation machine of Marmelstein. This combination is not a physical combination of the two inventions, but rather a combination of the teachings found in the reference. Motivation for combining a distributed architecture with an automatic code generation tool is readily found in the nature of the problem to be solved by the automatic code generation machine of Marmelstein as well as the knowledge of persons of ordinary skill in the art.

47. Regarding claim 37, Marmelstein teaches selecting packages for inclusion in the current model (column 17, line 52 – column 18, line 2). Specifically, when the user selects a package, the data structures and operations associated with that package appear in the current project window.

Additionally, Glebov teaches a common repository to store model objects. These objects can be imported into the current model (column 4, line 40 – column 5, line 23). It would have been obvious to a person of ordinary skill in the art at the time of Applicant's invention to combine the concept of importing model components as taught by Glebov with the automatic code generation machine of Marmelstein. This combination is not a physical combination of the two inventions, but rather a combination of the teachings found in the references. Motivation for combining the concept of importing model components with the automatic code generation machine of Marmelstein is readily found in the nature of the problem to be solved by an automatic code generation machine as well as the knowledge of persons of ordinary skill in the art.

48. Regarding claim 38, Marmelstein does not explicitly teach enforcing a set of business rules pertaining to a domain.

Glebov teaches modeling a business domain (Figs. 2, 5, and 6), specifically car ownership and insurance policies. Additionally, the Examiner considers this limitation one of intended use and respectfully requests that Applicant demonstrate how "enforcing a set of business rules pertaining to a domain" are patentably distinct from "enforcing a set of language rules pertaining to a programming language". In both fields of intended use, the graphical model must conform to a set of prescribed rules, those rules dictated by the intended use of the designer. It would have been obvious to a person of ordinary skill in the art at the time of Applicant's invention to combine the concept of modeling a business domain with the automatic code generation machine of Marmelstein because software designed with an automatic code generation machine must serve some utility, such as a business transaction. This combination is not a physical combination of the two inventions, but rather a combination of the teachings found in the references. Motivation for combining the concept of modeling a business domain with the automatic code generation machine of Marmelstein is readily found in the nature of the problem to be solved by an automatic code generation machine as well as the knowledge of persons of ordinary skill in the art.

Regarding claim 39, Marmelstein teaches generating a model (Fig. 6). It would have been obvious to a person of ordinary skill in the art at the time of Applicant's invention

that the model disclosed by Marmelstein is in accordance with a modeling language. Were it not so, the model could not convey its meaning and would cease to be useful.

Additionally, Glebov teaches generating a model in accordance with a modeling language, specifically Unified Modeling Language (column 4, lines 50-67). It would have been obvious to a person of ordinary skill in the art at the time of Applicant's invention to combine the concept of generating a model in a modeling language with the automatic code generation machine of Marmelstein, which generates models. This combination is not a physical combination of the two inventions, but rather a combination of the teachings found in the references. Motivation for combining the concept of generating a model using a modeling language with the automatic code generation machine of Marmelstein is readily found in the nature of the problem to be solved by an automatic code generation machine that generates models as well as the knowledge of persons of ordinary skill in the art.

49. Regarding claim 40, Marmelstein does not explicitly disclose generating a model in accordance with Unified Modeling Language.

Glebov teaches generating a model in accordance with a modeling language, specifically Unified Modeling Language (column 4, lines 50-67). It would have been obvious to a person of ordinary skill in the art at the time of Applicant's invention to combine the concept of generating a model in a Unified Modeling Language with the automatic code generation machine of Marmelstein, which generates models. This combination is not a physical combination of the two inventions, but rather a

combination of the teachings found in the references. Motivation for combining the concept of generating a model using a Unified Modeling Language with the automatic code generation machine of Marmelstein is readily found in the nature of the problem to be solved by an automatic code generation machine that generates models as well as the knowledge of persons of ordinary skill in the art.

Conclusion

Applicant's amendment necessitated the new ground(s) of rejection presented in this Office action. Accordingly, **THIS ACTION IS MADE FINAL**. See MPEP § 706.07(a). Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the date of this final action.

Art considered pertinent by the examiner but not applied has been cited on form PTO-892. An updated search has revealed new prior art, which has been cited.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Jason Proctor whose telephone number is (571) 272-3713. The examiner can normally be reached on 8:30 am-4:30 pm M-F.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Kevin J Teska can be reached on (571) 272-3716. The fax phone number for the organization where this application or proceeding is assigned is 703-872-9306.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

jsp

Jason Proctor
Examiner
Art Unit 2123

KEVIN J. TESKA
SUPERVISORY
PATENT EXAMINER